Entanglement Electro Lab

Amrit Singh & Nived Nandakumar

Period 7 Electro

Plots

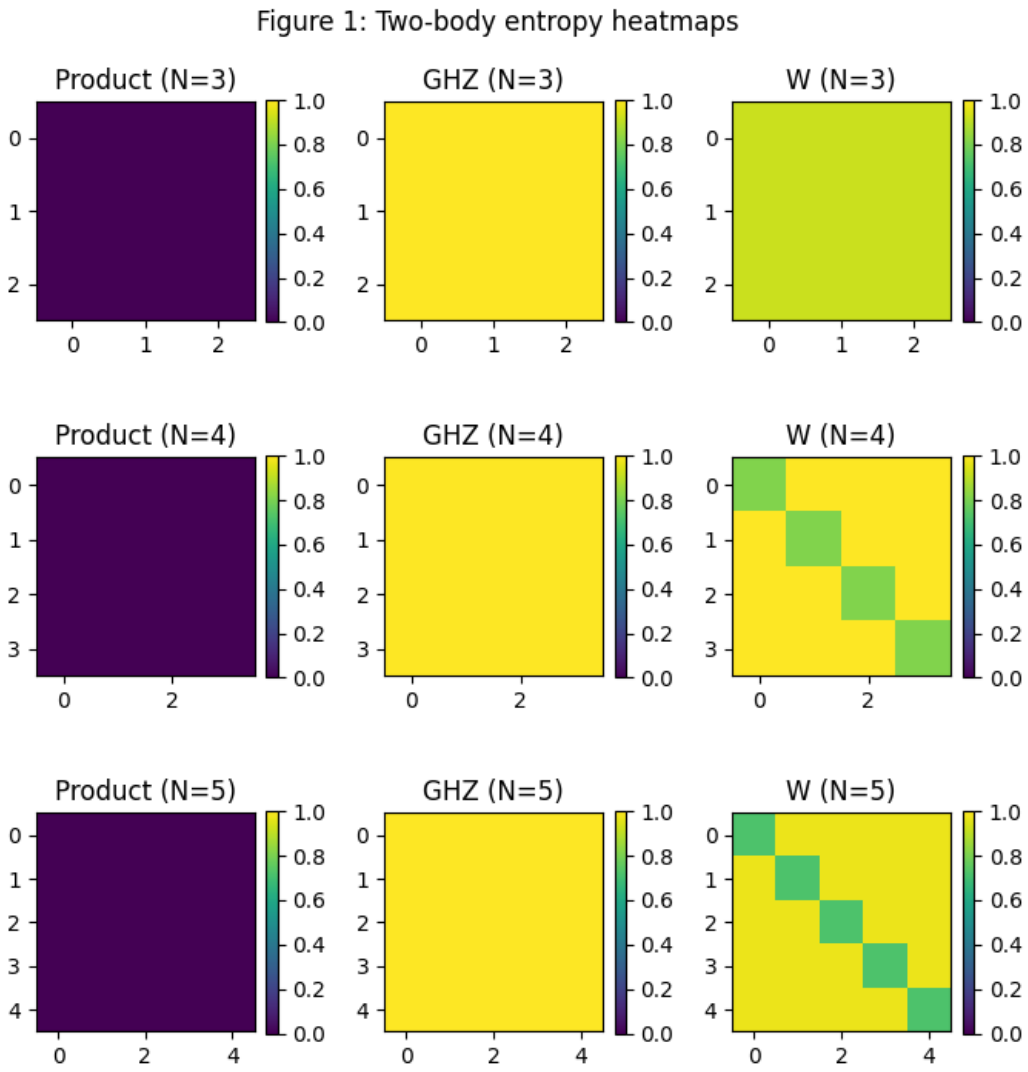

Figure 1: Two-body entropy heatmaps for product, GHZ, and W states (N = 3, 4, 5).

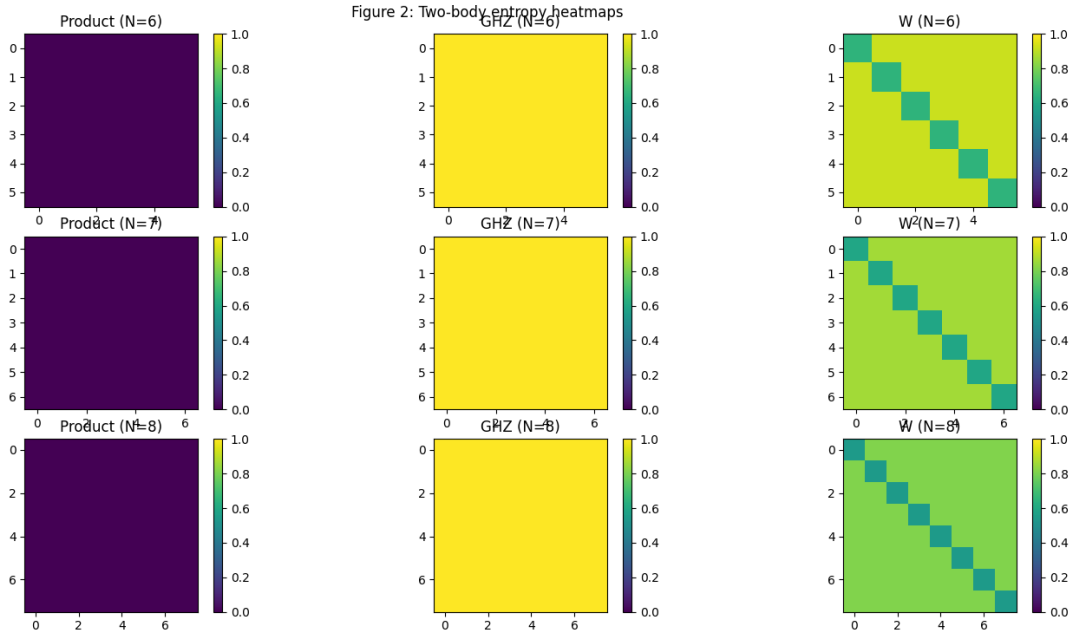

Figure 2: Two-body entropy heatmaps for product, GHZ, and W states (N = 6, 7, 8).

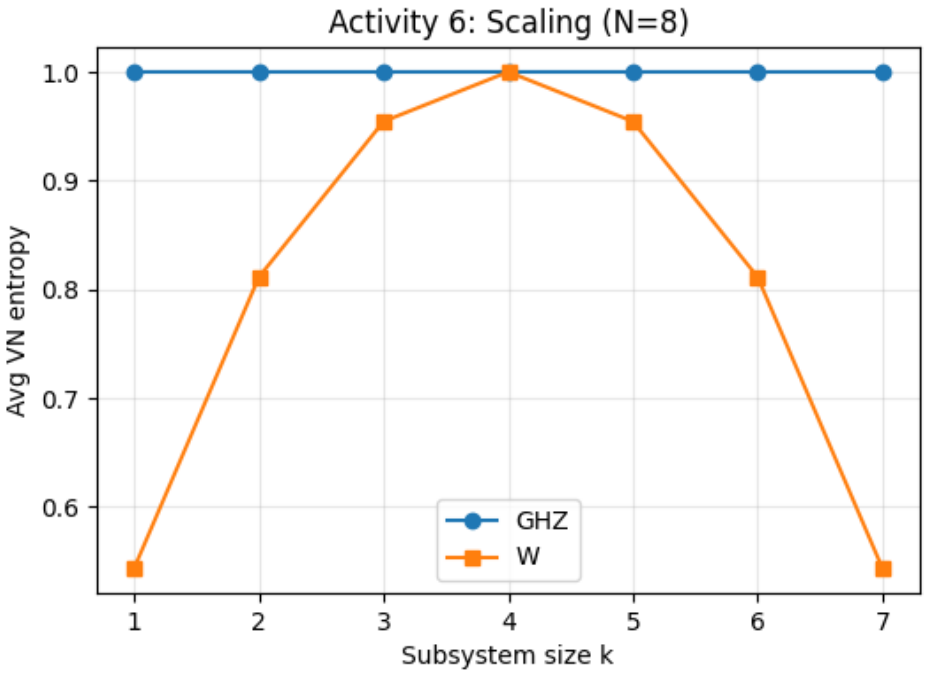

Figure 3: Average entropy vs. subsystem size k for GHZ and W states with $N = 8$.

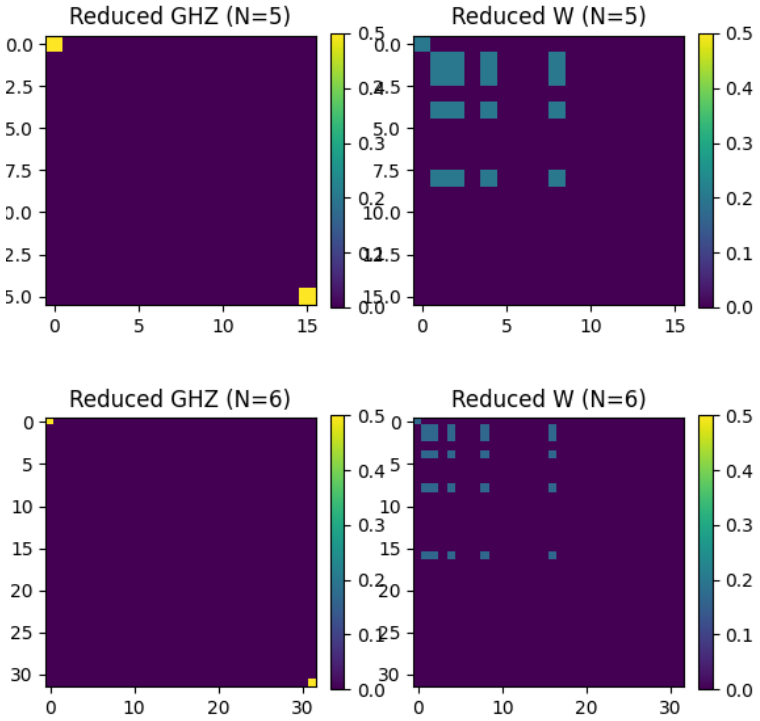

Figure 4: Reduced density matrix heatmaps for GHZ and W states after removing one qubit (N = 5 and N = 6).

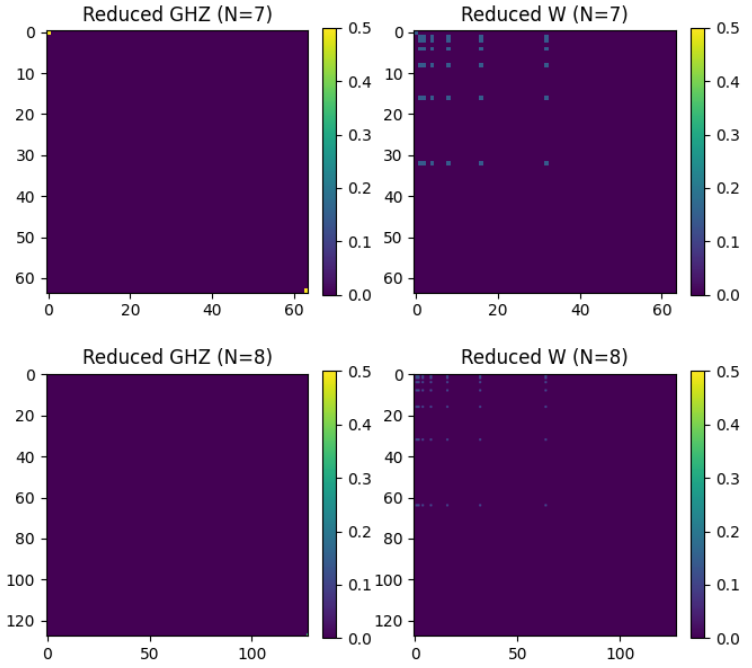

Figure 5: Reduced density matrix heatmaps for GHZ and W states after removing one qubit (N = 7 and N = 8).

Required Questions

1. Amount vs. Structure of Entanglement

The amount of entanglement refers to how mixed a subsystem is, quantified by numbers such as von Neumann entropy or purity. These single values indicate how strongly a subsystem is entangled with the rest of the system. In contrast, the structure of entanglement describes how this entanglement is distributed across particles and subsystems. Heatmaps and bipartition plots reveal whether correlations are global (as in GHZ states) or distributed across overlapping subsystems (as in W states).

2. Example of Similar Amount but Different Structure

For $N = 8$, both GHZ and W states reach comparable entropy near $k = 4$ in the Average Entropy vs. k plot, suggesting similar total entanglement. However, their heatmaps differ dramatically: GHZ shows uniform maximal entropy everywhere, while W exhibits structured patterns with reduced diagonal values. Thus, although the amount of entanglement can appear similar numerically, the spatial distribution is fundamentally different.

3. Symmetry in GHZ and W States

Both GHZ_N and W_N states are symmetric under qubit permutations, which appears as uniform rows and columns in the entropy heatmaps and symmetric Average Entropy vs. k curves about $k = N/2$. This symmetry means all qubits play equivalent roles. It simplifies interpretation because averaging over subsets does not depend on which qubits are chosen, allowing us to focus on subsystem size rather than specific configurations.

4. Global vs. Local Coherence and Robustness

Global coherence refers to quantum coherence stored only in the full N-qubit system. Local (or internal) coherence survives within smaller subsystems.

GHZ states rely primarily on global coherence: their reduced density matrices show coherence only at extreme corners, which disappears once a qubit is removed. W states rely on local/internal coherence: after particle loss, off-diagonal structure remains across many entries, indicating surviving entanglement.

This explains robustness: removing one qubit from a GHZ state collapses it into a classical mixture, while W states retain distributed coherence and remain entangled. Consequently, W states are far more robust under particle loss than GHZ states.

Appendix: Python Code

```
import os
import numpy as np
import matplotlib.pyplot as plt
from itertools import combinations
from qutip import basis, tensor, ket2dm, ptrace, entropy_vn

np.set_printoptions(precision=4, suppress=True)

SAVE_PLOTS = False
OUTDIR = "lab_outputs"
if SAVE_PLOTS:
    os.makedirs(OUTDIR, exist_ok=True)

zero = basis(2,0)
one  = basis(2,1)

def purity(rho):
    return rho.purity()

def zero_state(N):
    return tensor([zero]*N)

def one_state(N):
    return tensor([one]*N)

def vn_entropy_subset(rho, keep):
    return entropy_vn(ptrace(rho, keep), base=2)

def ghz_state(N):
    return (zero_state(N) + one_state(N)).unit()

def w_state(N):
    terms=[]
    for i in range(N):
        k=[zero]*N
        k[i]=one
        terms.append(tensor(k))
    return sum(terms).unit()

def entropy_map_2body(rho,N):
    M=np.zeros((N,N))
    for i in range(N):
        M[i,i]=vn_entropy_subset(rho,[i])
    for i in range(N):
        for j in range(N):
            if i!=j:
                M[i,j]=vn_entropy_subset(rho,[i,j])
    return M

def plot_entropy_grid(N_list,title):
    fig,axs=plt.subplots(len(N_list),3,figsize=(9,9))
    for r,N in enumerate(N_list):
```

```
        states=[
            ("Product", ket2dm(zero_state(N))),
            ("GHZ", ket2dm(ghz_state(N))),
            ("W", ket2dm(w_state(N)))
        ]
        for c,(name,rho) in enumerate(states):
            M=entropy_map_2body(rho,N)
            im=axs[r,c].imshow(M, vmin=0.0, vmax=1.0)
            axs[r,c].set_title(f"{name} (N={N})")
            plt.colorbar(im,ax=axs[r,c],fraction=0.046)
    plt.tight_layout()
    plt.show()

def avg_entropy_for_k(rho,N,k):
    vals=[]
    for keep in combinations(range(N),k):
        vals.append(vn_entropy_subset(rho,list(keep)))
    return np.mean(vals)

def plot_avg_entropy_vs_k():
    N=8
    rho_g=ket2dm(ghz_state(N))
    rho_w=ket2dm(w_state(N))
    ks=range(1,N)
    g=[avg_entropy_for_k(rho_g,N,k) for k in ks]
    w=[avg_entropy_for_k(rho_w,N,k) for k in ks]
    plt.plot(ks,g,'o-',label="GHZ")
    plt.plot(ks,w,'s-',label="W")
    plt.legend()
    plt.show()

def remove_one_qubit(rho,N):
    return ptrace(rho,list(range(1,N)))

def plot_reduced_density():
    for Ns in [(5,6),(7,8)]:
        fig,axs=plt.subplots(2,2,figsize=(8,8))
        for r,N in enumerate(Ns):
            rho_g=remove_one_qubit(ket2dm(ghz_state(N)),N)
            rho_w=remove_one_qubit(ket2dm(w_state(N)),N)
            axs[r,0].imshow(np.abs(rho_g.full()),vmin=0,vmax=.5)
            axs[r,1].imshow(np.abs(rho_w.full()),vmin=0,vmax=.5)
        plt.show()
```